Insper

Árvores, Bagging e Random Forest

Aula 5

Magno Severino
PADS - Modelos Preditivos
28/05/2021

Na aula passada...

- Definição de um problema de classificação;
- Regressão Logística;
- kNN para classificação;
- Árvores de classificação;
- Métricas para avaliação de modelos de classificação (acurácia, sensibilidade, especificidade, entre outras).

Objetivos de aprendizagem

Ao final dessa aula você deverá ser capaz de

- conceituar uma árvore de classificação/regressão;
- compreender as limitações de uma árvore;
- conceituar *bagging*;
- conceituar floresta aleatória.

Árvores de regressão

Lembre-se que em um problema de regressão, a variável resposta é quantitativa, $Y \in \mathbb{R}$.

O algoritmo CART constrói a árvore de regressão de maneira análoga ao caso de classificação.

A principal diferença é que, para definir as divisões, utilizamos uma perda quadrática ao invés do erro de classificação

$$(\hat{j}, \hat{t}\,) = \arg\min_{(j,t)} \left\{ \sum_{\{i:x_i \in R_1\}} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{\{i:x_i \in R_2\}} \left(y_i - \hat{y}_{R_2}\right)^2 \right\},$$

em que $\hat{y}_{R_1}$ e $\hat{y}_{R_2}$ são as médias das respostas dos dados de treinamento que pertencem às regiões R_1 e R_2, respectivamente.

Árvores de regressão

Para cada região R_j correspondente a um nó terminal da árvore de regressão obtida, o CART associa uma constante c_j que é a média das respostas dos dados de treinamento que pertencem à região R_j,

$$c_j = \frac{1}{n_j} \sum_{\{i:x_i \in R_j\}} y_i.$$

Então, a estimativa CART para a função de regressão é

$$\hat{f}\left(x\right) = \sum_j c_j \mathbb{I}_{R_j}(x).$$

Vantagens e desvantagens

Aspectos Positivos

- Fácil de explicar (muito mais que regressão linear);
- Podem ser apresentadas graficamente e facilmente interpretadas por pessoas que não são especialistas no assunto;
- Tratam facilmente preditores qualitativos, sem a necessidade da criação de variáveis indicadoras / *dummies*;
- Não é sensível à escala como outros métodos.

Aspectos Negativos

- Uma pequena alteração nos dados pode causar uma grande alteração na árvore estimada (variância alta);
- Previsões baseadas em regiões retangulares;
- Não apresentam desempenho preditivo tão bom quanto outros métodos.

Vantagens e desvantagens

Aspectos Positivos

- Fácil de explicar (muito mais que regressão linear);
- Podem ser apresentadas graficamente e facilmente interpretadas por pessoas que não são especialistas no assunto;
- Tratam facilmente preditores qualitativos, sem a necessidade da criação de variáveis indicadoras / *dummies*;
- Não é sensível à escala como outros métodos.

Aspectos Negativos

- **Uma pequena alteração nos dados pode causar uma grande alteração na árvore estimada (variância alta);**
- Previsões baseadas em regiões retangulares;
- **Não apresentam desempenho preditivo tão bom quanto outros métodos.**

Bootstrap aggregation (bagging)

Árvores

- Sabemos como crescer árvores de classificação e regressão.
- A vantagem das árvores de classificação e regressão é a sua interpretabilidade.
- As árvores resultantes do processo de crescimento podem ser exibidas graficamente, e a regra decisão correspondente é interpretável até mesmo por um não especialista.
- Em uma árvore, variáveis preditoras qualitativas são tratadas diretamente (as divisões são feitas considerando-se a separação dos possíveis valores da preditora em dois conjuntos disjuntos) sem a necessidade de se criar variáveis dummy.
- Em geral, as árvores não tem boa performance preditiva.
- Além disso, árvores não são muito robustas: uma pequena mudança nos dados de treinamento leva a uma grande mudança na árvore crescida. Ou seja, algoritmos como o CART sofrem por ter **variância alta**.
- Agregando-se muitas árvores de decisão com métodos como o *bagging* (agregação bootstrap), a performance preditiva das árvores pode ser melhorada.

Lembre-se

$$\text{Erro de predição} = \text{viés}^2 + \text{variância} + (\text{erro irredutível}).$$

- Problema com as árvores: variância alta.
- Vamos apresentar na sequencia uma alternativa que tem por objetivo de reduzir a variância.
- Sejam $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_k$ k regressores obtidos ao treinar um mesmo modelo em k conjuntos de dados distindos. Seja $\bar{f} = \frac{1}{n}\sum_{i=1}^{k} \hat{f}_i$. Então,

$$\text{Var}(\bar{f}) = \frac{1}{n^2}\sum_{i=1}^{k} \text{Var}(\hat{f}_i) + \frac{1}{n^2}\sum_{i \neq j} \text{Cov}(\hat{f}_i, \hat{f}_j).$$

- Na prática temos apenas **um** conjunto de dados disponível.
- Uma maneira de obter contornar este problema é através do método *boostrap*.

Bootstrap

Considere que temos o seguinte conjunto de dados observados.

ID	Idade	Genero	Salario
1	20	F	19.04
2	44	F	13.25
3	34	M	21.97
4	41	NA	22.88
5	32	M	27.04

Bootstrap

A ideia do boostrap é gerar B amostras **com reposição** a partir dos dados observados

Amostra original			
ID	**Idade**	**Genero**	**Salario**
1	20	F	19.04
2	44	F	13.25
3	34	M	21.97
4	41	NA	22.88
5	32	M	27.04

↓

Amostra bootstrap 1			
ID	**Idade**	**Genero**	**Salario**
2	44	F	13.25
2	44	F	13.25
5	32	M	27.04
4	41	NA	22.88
1	20	F	19.04

Amostra bootstrap 2			
ID	**Idade**	**Genero**	**Salario**
2	44	F	13.25
3	34	M	21.97
5	32	M	27.04
4	41	NA	22.88
5	32	M	27.04

⋮

Amostra bootstrap B			
ID	**Idade**	**Genero**	**Salario**
5	32	M	27.04
1	20	F	19.04
2	44	F	13.25
2	44	F	13.25
4	41	NA	22.88

Bootstrap aggregation (bagging)

- A agregação *bootstrap*, ou *bagging*, foi proposta por Leo Breiman em 1996.
- O bagging é um procedimento geral, não restrito apenas ao contexto de árvores de classificação e regressão, cujo propósito é reduzir a variância de um método de aprendizagem.
- Uma maneira natural de se reduzir a variância de um método de aprendizagem seria tomar muitos conjuntos de treinamento e com cada um deles treinar regressores, cujas respostas seriam tomadas em média como sendo a resposta resultante do conjunto (*ensemble*) de regressores (daí o termo "*método de ensemble*").
- Este caminho não é viável, pois em geral não temos acesso a múltiplos conjuntos de dados de treinamento.

Bagging

- A ideia fundamental do *bagging* é gerar amostras bootstrap a partir do conjunto original de dados de treinamento.
- No *bagging*, em um problema de regressão, geramos B amostras bootstrap que farão o papel de dados de treinamento em cada regressor treinado.
- Assim, treinamos os regressores $\hat{f}_1, \ldots, \hat{f}_B$ e o regressor agregado é a média

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{i=1}^{B} \hat{f}_i(x).$$

- Quando os regressores são treinados via algoritmo CART, não podamos as árvores resultantes: crescemos árvores bem altas, que, individualmente, terão grande variância e viés baixo.
- O processo de *bagging* cuida da redução da variância.

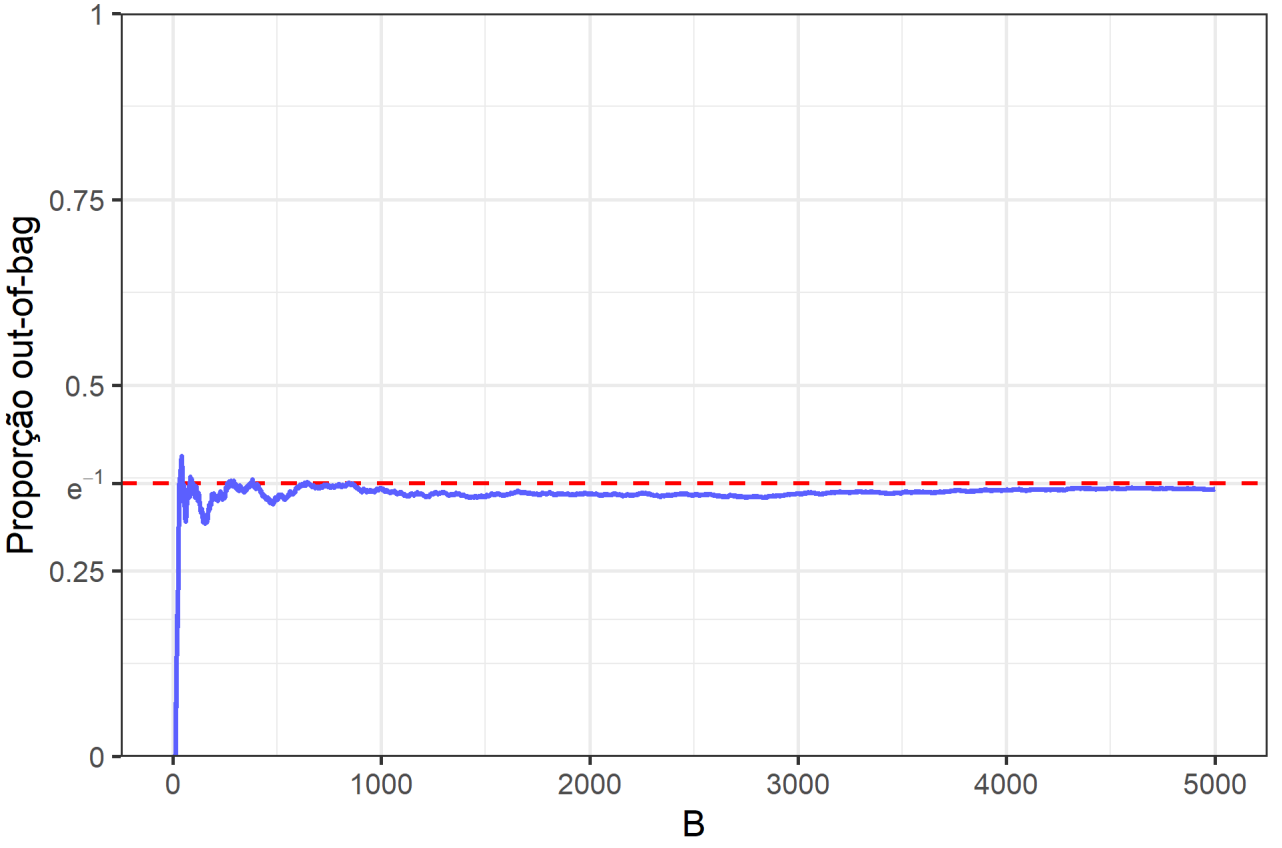
Bagging

- Quando temos árvores de classificação, o *bagging* funciona de maneira análoga.
- A diferença é que o classificador agregado $\hat{f}_{\text{bag}}$ prevê segundo o "voto da maioria" dos classificadores $\hat{f}_1, \ldots, \hat{f}_B$ treinados a partir das B amostras bootstrap.
- O bagging produz grandes ganhos de performance preditiva quando agregamos várias árvores, formando um único regressor ou classificador.

Erro *out-of-bag*

- O bagging fornece uma maneira simples e barata de se estimar o erro de teste do regressor (ou classificador) agregado sem precisarmos recorrer a procedimentos de validação cruzada.
- É possível mostrar que, em média, cada árvore crescida durante o procedimento do bagging utiliza aproximadamente dois terços dos dados originais de treinamento.
- Os dados do um terço remanescente de uma determinada árvore do ensemble são denominados observações *out-of-bag* (fora da sacola) da árvore em questão.
- Podemos prever a resposta do i-ésimo dado de treinamento utilizando todas as árvores do ensemble nas quais este dado pertence às observações *out-of-bag*.
- Para tanto, tomamos a média das respostas previstas para este dado (em um problema de regressão), ou o voto da maioria (em um problema de classificação), das árvores do *ensemble* em que o i-ésimo dado ficou fora da sacola.
- Fazendo isto para cada um dos dados de treinamento obtemos uma estimativa válida do erro de teste esperado.
- Quando temos muitos dados de treinamento este procedimento é vantajoso, pois a validação cruzada nestes casos seria extremamente custosa, em termos computacionais.

Erro *out-of-bag*



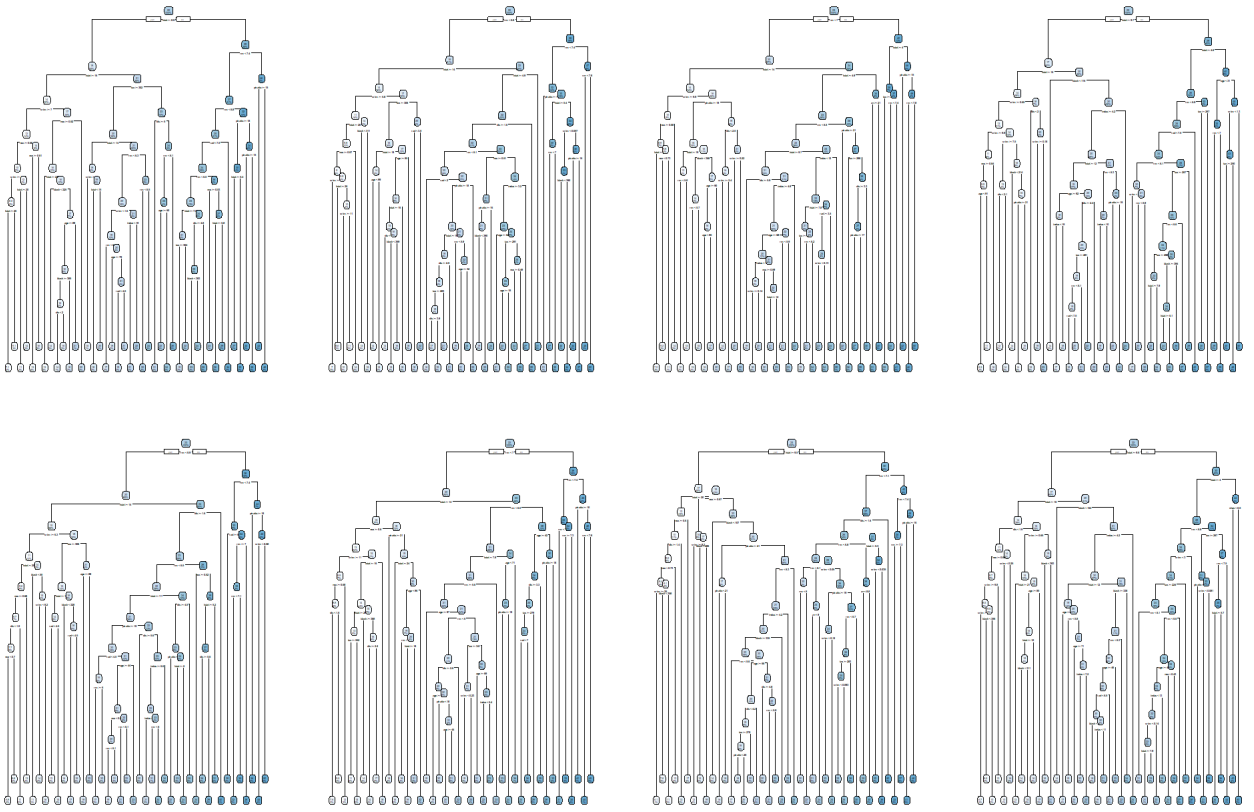
$e^{-1} \approx 0{,}368.$

Housing Values in Suburbs of Boston

Dados do pacote `MASS`, com as seguintes informações:

- **crim**: per capita crime rate by town.
- **zn**: proportion of residential land zoned for lots over 25,000 sq.ft.
- **indus**: proportion of non-retail business acres per town.
- **chas**: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- **nox**: nitrogen oxides concentration (parts per 10 million).
- **rm**: average number of rooms per dwelling.
- **age**: proportion of owner-occupied units built prior to 1940.
- **dis**: weighted mean of distances to five Boston employment centres.
- **rad**: index of accessibility to radial highways.
- **tax**: full-value property-tax rate per $10,000.
- **ptratio**: pupil-teacher ratio by town.
- **black**: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
- **lstat**: lower status of the population (percent).
- **medv**: median value of owner-occupied homes in $1000s.

Bagging



Qual é um possível problema com essas árvores?

Floresta aleatória

Florestas aleatória

1. Para $b = 1, \ldots, B$, faça:

    1. Obtenha uma amostra bootstrap $\mathbf{Z}^*$ de tamanho N do conjunto de treinamento.

    2. Cresça uma árvore T_b a partir da amostra bootstrap, repetindo recursivamente os seguintes passos para cada nó terminal da árvore até alcançar o número mínimo de observações n_{min} de cada nó.

        a. Selecione aleatoriamente m variáveis dentre as p variáveis disponíveis.

        b. Escolha a melhor variável para a divisão/split entre as m escolhidas.

        c. Divida o nó em dois nós descendentes.

2. Retorne o comitê de árvores $\{T_b\}_{b=1}^{B}$.

Para fazer a previsão pra uma observação x:

- **Regressão:** $\hat{f}^B = \frac{1}{B}\sum_{i=1}^{B} T_b(x)$.

- **Classificação:** Seja $\hat{C}_b^B(x)$ a classe predita pela b-ésimia árvore. Então, $\hat{C}^B(x) = \text{voto da maioria}\{\hat{C}_b^B(x)\}_{i=1}^{B}$.

Bagging

Split 1	Split 2	...	Split S
crim	crim	...	crim
zn	zn	...	zn
indus	indus	...	indus
chas	chas	...	chas
nox	nox	...	nox
rm	rm	...	rm
age	age	...	age
dis	dis	...	dis
rad	rad	...	rad
tax	tax	...	tax
ptratio	ptratio	...	ptratio
black	black	...	black
lstat	lstat	...	lstat

Random forest

Split 1	Split 2	...	Split S
crim	crim	...	crim
zn	zn	...	zn
indus	indus	...	indus
chas	chas	...	chas
nox	nox	...	nox
rm	rm	...	rm
age	age	...	age
dis	dis	...	dis
rad	rad	...	rad
tax	tax	...	tax
ptratio	ptratio	...	ptratio
black	black	...	black
lstat	lstat	...	lstat

Florestas aleatórias - hiperparâmetros

Número de preditoras a ser considerado em cada divisão/split

- Para **classificação** recomenda-se utilizar $\sqrt{p}$ e número mínimo de observações por nó igual a 1.
- Já para **regressão** recomenda-se utilizar $\frac{p}{3}$ e número mínimo de observações por nó igual a 5.

Número de árvores da floresta

Tipicamente se utiliza de 500 a 1000 árvores. É importante notar que, geralmente, aumentar B não causa sobreajuste dos dados.

Dados Boston

```
library(randomForest)
library(MASS)
data(Boston)
set.seed(123)

(rf <- randomForest(medv ~ ., data = Boston))
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = Boston)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 9.755933
##                     % Var explained: 88.44
```

Mean of squared residuals: `MSE = mean((Boston$medv - rf$predicted)^2)`

% Var explained: `1 - MSE/var(Boston$medv)`

Dados Boston

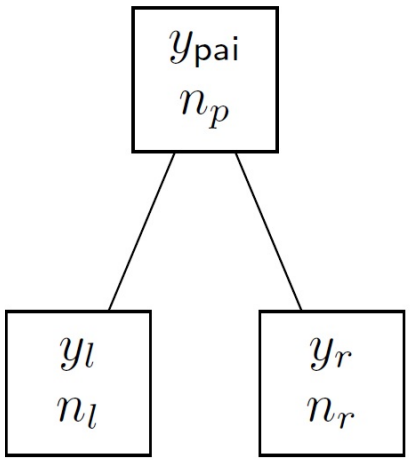
```
tibble(arvore = 1:length(rf$mse), mse = rf$mse) %>%
  ggplot(aes(arvore, mse)) +
  geom_line(color = "#66A61E", size = 1.2) +
  labs(x="Número de Árvores", y="MSE (OOB)") +
  theme_bw()
```

Prática R

Tarefa: tentar diferentes números de variáveis para serem consideradas a cada split

Importância de variáveis

Verifica-se a redução total no RSS (*residual sum of squares*) devido a divisão/split relativa a uma dada preditora para cada árvore e calcula-se a média dessas reduções de acordo com número de árvores na floresta. Portanto, um valor alto indica que a preditora/variável é importante.

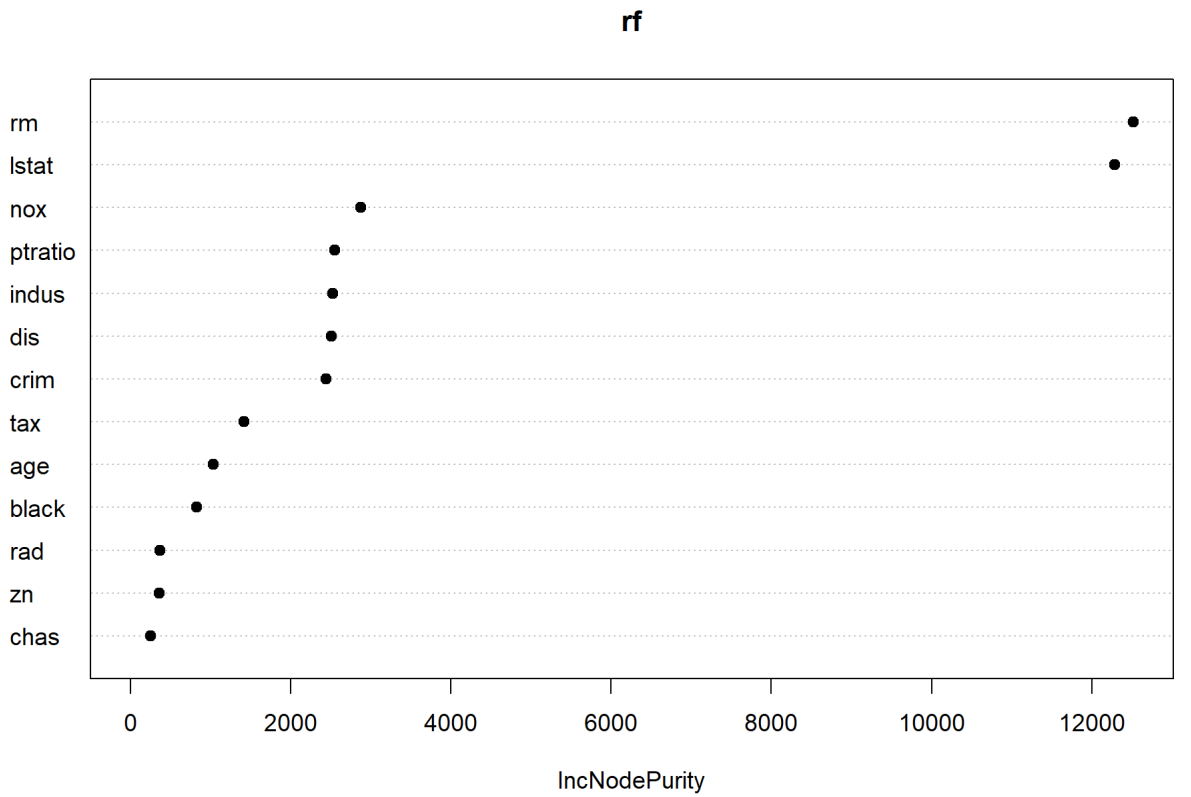


$$\text{RSS}_{\text{pai}} - \left(\text{RSS}_{\text{l}} + \text{RSS}_{\text{r}}\right)$$

$$\sum_{i=1}^{n_p}(y_{\text{pai},i} - \bar{y}_{\text{pai},i})^2 - \left(\sum_{i=1}^{n_l}(y_{\text{l},i} - \bar{y}_{\text{l},i})^2 + \sum_{i=1}(y_{\text{r},i} - \bar{y}_{\text{r},i})^2\right)$$

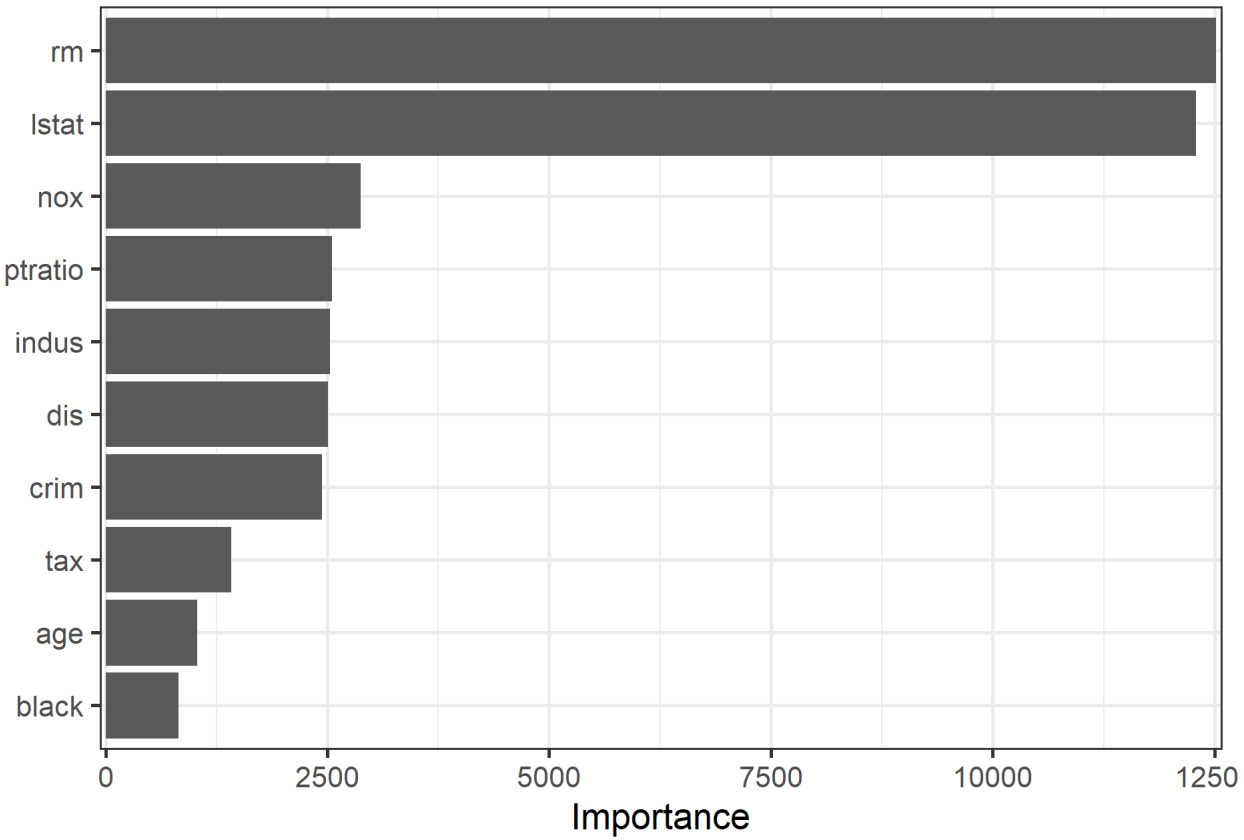
Importância de variáveis

```
randomForest::varImpPlot(rf, pch = 19)
```



Importância de variáveis - alternativa

```
vip::vip(rf)
```



Churn

Dados do pacote `modeldata` contendo 5.000 observações e 19 preditoras para churn:

- state
- account_length
- area_code
- international_plan
- voice_mail_plan
- number_vmail_messages
- total_day_minutes
- total_day_calls
- total_day_charge
- total_eve_minutes
- total_eve_calls
- total_eve_charge
- total_night_minutes
- total_night_calls
- total_night_charge
- total_intl_minutes
- total_intl_calls
- total_intl_charge
- number_customer_service_calls
- churn

Dados Churn - floresta aleatória

```
library(modeldata)
library(randomForest)
library(rsample)

data(mlc_churn)

mlc_churn$churn <- factor(mlc_churn$churn,
                          levels = c("no", "yes"))

set.seed(123)

splits <- initial_split(mlc_churn, prop = .9, strata = "churn")

treino <- training(splits); teste <- testing(splits)

(rf <- randomForest(churn ~ ., data = treino))
```

Dados Churn - floresta aleatória

```
##
## Call:
##  randomForest(formula = churn ~ ., data = treino)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 4.27%
## Confusion matrix:
##       no yes class.error
## no  3819  45  0.01164596
## yes  147 490  0.23076923
```

A matriz de confusão é baseada nos dados OOB. As linhas da matriz indicam a categoria da resposta observada e as colunas indicam as categorias classificadas.

Dados Churn

```
tibble(arvore = 1:nrow(rf$err.rate),
       oob = rf$err.rate[,1]) %>%
  ggplot(aes(arvore, oob)) +
  geom_line(color = "#66A61E", size = 1.2) +
  labs(x = "Número de Árvores", y = "Erro de Classificação (OOB)")
  theme_bw()
```

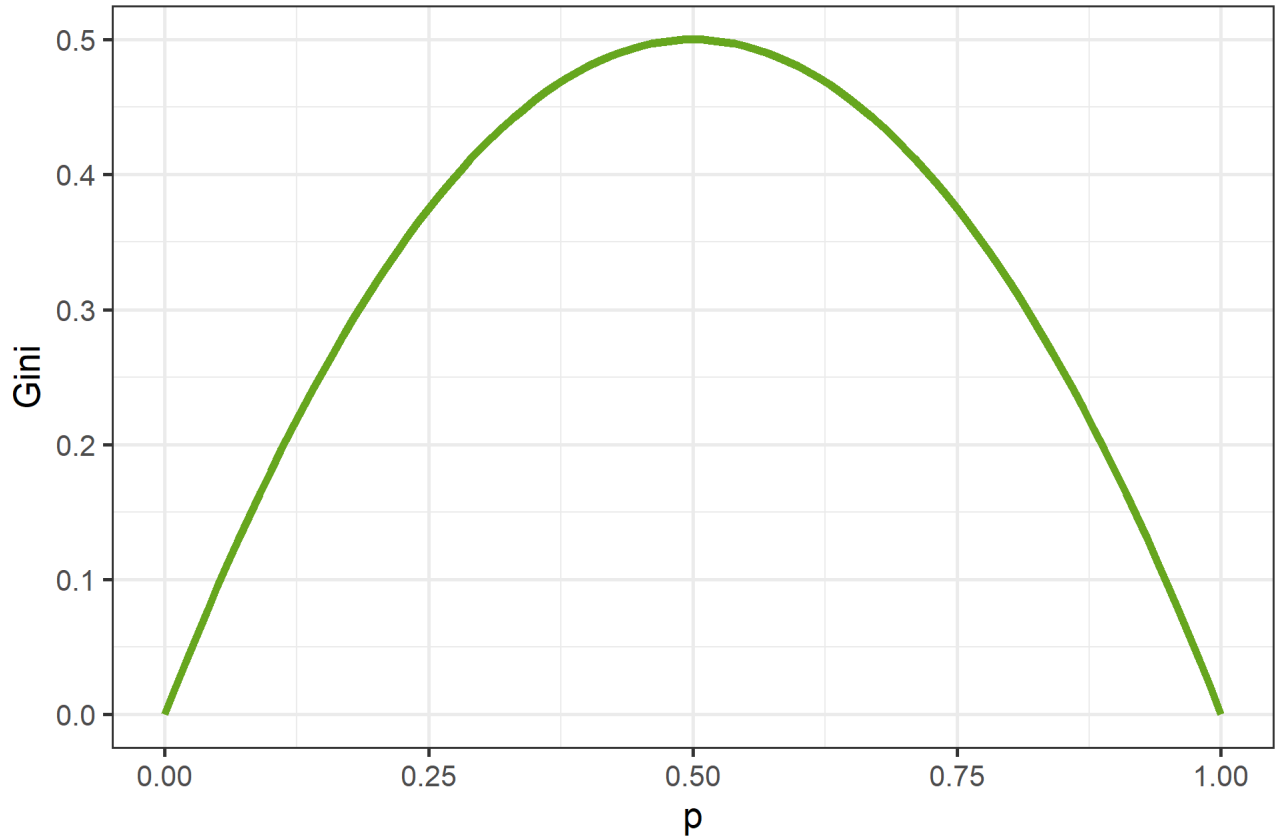
Prática R

Tarefa: tentar diferentes números de variáveis para serem consideradas a cada split

Gini

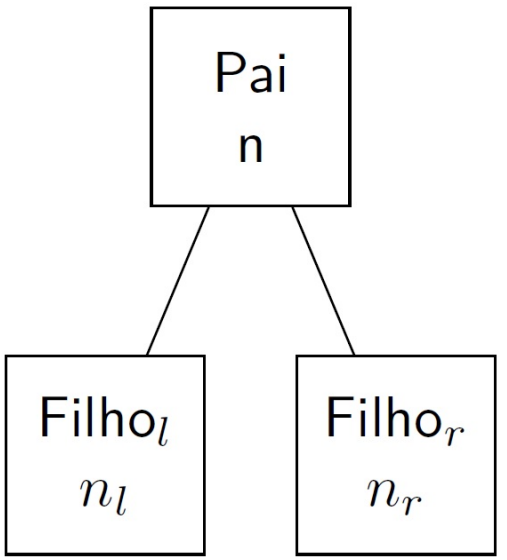
Seja C o conjunto das classes da variável resposta.

$$\text{Gini}(\text{nó}) = \sum_{i \in C} p_i(1 - p_i)$$



Importância de variáveis

Verifica-se a redução total no índice de Gini para cada preditora com as observações utilizadas na construção da árvore (*in bag*) e calcula-se a média de acordo com o número de árvores na floresta. A redução é ponderada pelo número de observações no nó ancestral e descendentes.

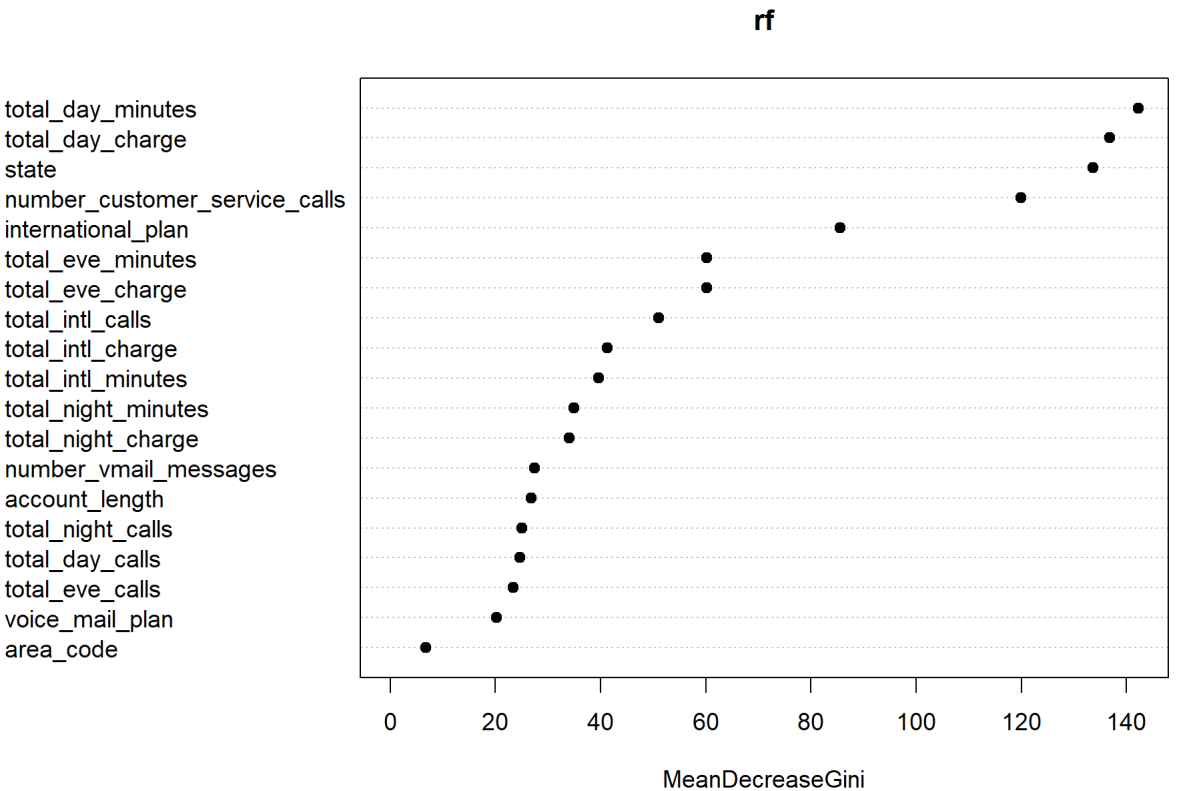


$$n \times \text{Gini}(\text{pai}) - \Big(n_l \times \text{Gini}(\text{Filho}_\text{l}) + n_r \times \text{Gini}(\text{Filho}_\text{r})\Big)$$

Importância de variáveis

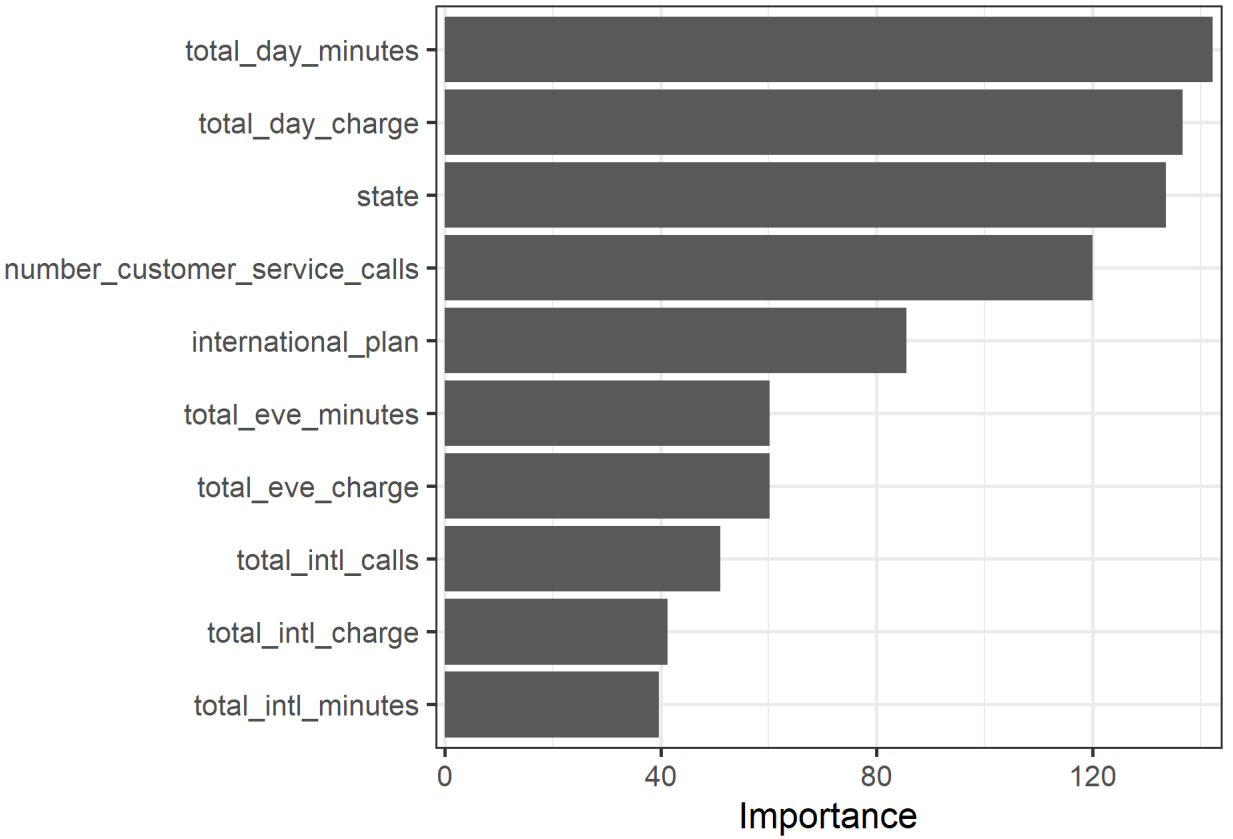
```
rf <- randomForest(churn ~ ., data = treino)

varImpPlot(rf, pch = 19)
```



Importância de variáveis - alternativa

```
vip::vip(rf)
```



Dados Churn - predição

```
head(predict(rf, teste, type = "prob"))
```

```
##      no   yes
## 1 0.988 0.012
## 2 0.944 0.056
## 3 0.964 0.036
## 4 0.878 0.122
## 5 0.994 0.006
## 6 0.960 0.040
```

```
head(predict(rf, teste))
```

```
##  1  2  3  4  5  6
## no no no no no no
## Levels: no yes
```

Dados Churn - matriz de confusão

```
library(pROC)

pred_rf <- predict(rf, teste)

table(observado = teste$churn, predito = pred_rf)
```

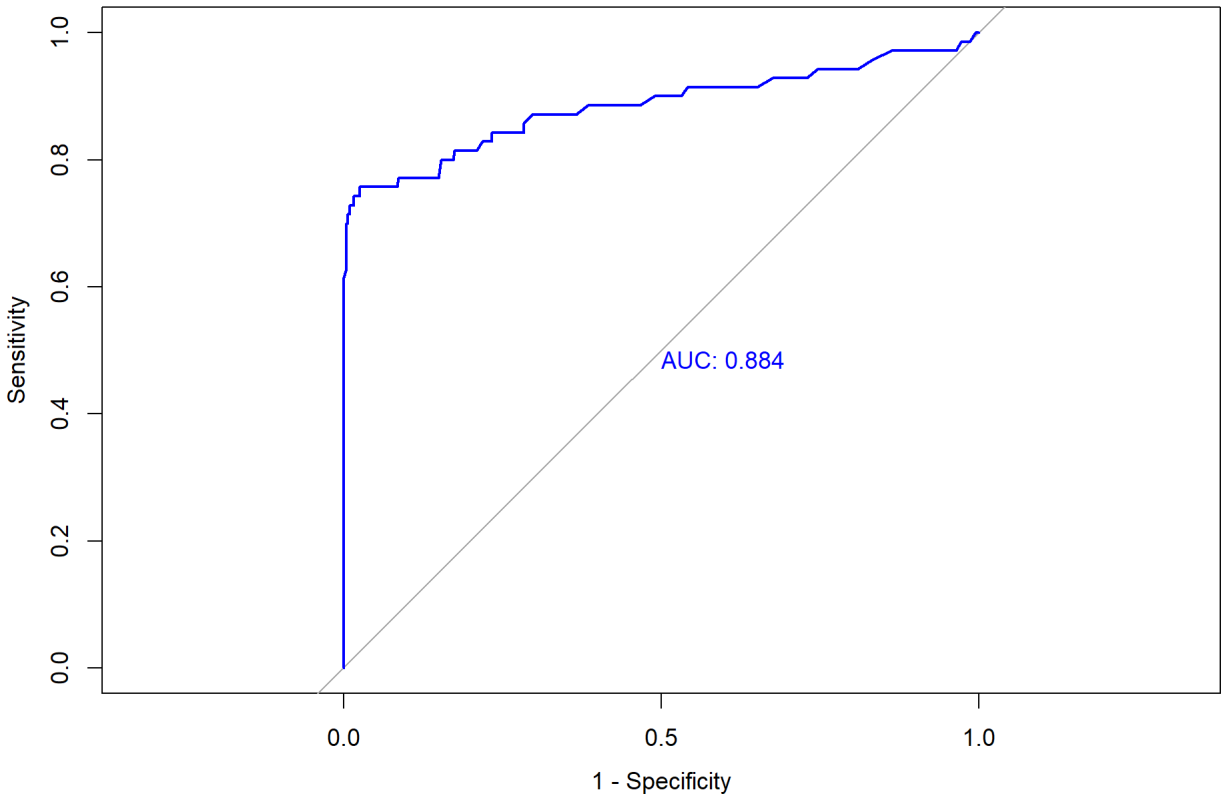
```
##           predito
## observado  no yes
##       no  427   2
##       yes  22  48
```

```
desempenho <- roc(teste$churn,
                  predict(rf, teste, type = "prob")[,2])
coords(desempenho, .5,
       ret = c("1-accuracy", "sensitivity",
               "specificity", "ppv", "npv"))
```

```
##           1-accuracy sensitivity specificity  ppv       npv
## threshold 0.04809619   0.6857143    0.995338 0.96 0.9510022
```

Dados Churn - curva ROC

```
roc(teste$churn, predict(rf, teste, type = "prob")[,2],
    plot = TRUE, print.auc=TRUE, col = "blue", legacy.axes = TRUE)
```



##

O pacote `ranger`

O pacote `ranger` apresenta um excelente desempenho computacional para analise bases de dados com maior volume.

```
library(ranger)

ranger(churn ~ ., data = treino)
```

```
## Ranger result
##
## Call:
##  ranger(churn ~ ., data = treino)
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      4501
## Number of independent variables:  19
## Mtry:                             4
## Target node size:                 1
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error:             4.04 %
```

Resumindo...

- Árvore de classificação/regressão sofre de variância alta.
- Alternativas:
  - **Bootstrap aggregation**: gera B amostras bootstrap e uma árvore para cada amostra.
  - **Random forest**: similar ao *bagging*, entretanto, considera apenas uma amostra das preditoras a cada divisão das árvores.

Obrigado!

magnotfs@insper.edu.br